# FAST FINGERS

by

Craig Chamberlain

A clever new idea in autobooting programs for both disk and cassette. Turning on your computer was never so much fun!

Just imagine - you are sitting in front of your ATARI, you pop in the ATARI BASIC cartridge and turn the computer on, it makes the usual weird sounds, and it says "READY". But then it says "OR NOT, HERE I COME!", which produces a BASIC error. It follows this with more commands to BASIC which are immediately executed. It even starts messing around with your cassette or disk drive, LOADing and RUNning files. And all the while, every time you see each character mysteriously appear on the screen, you hear the associated keyboard click. It's as if a ghost was pounding away at your keyboard.

Has it finally happened? Has your computer gone absolutely mad? The day you always feared, when your computer would take over and replace YOU, has it arrived? No. Relax. You're just the innocent victim of the ultimate autoboot program, called FAST FINGERS.

## DISK AND CASSETTE AUTOBOOT PROGRAMS

An autobooting program is one which is automatically executed when the computer is powered up. The program could be a game which immediately starts playing, or perhaps just a little utility to configure the system before use. An example of such a utility would be one that changed the margins before giving control to the user.

If you have a disk based system and use DOS II, you are probably aware that this autoboot feature is available with the AUTORUN.SYS file. This is a binary file which will execute before the system starts the cartridge or calls DUP. In order to create a binary file which will RUN a BASIC program, a separate program, a boot file maker, must be used. The boot file maker generates a binary file which, in turn, instructs the computer to RUN the BASIC program.

Although most people are not aware of this, the cassette handler also has a method of performing an autoboot. If the START key is depressed while the computer is turned on, a cassette file will be loaded and executed before the computer enters BASIC. Again, this file must be in binary form.

## FAST FINGERS - THE ULTIMATE AUTOBOOT

Many autoboot file making programs have appeared, but most of them have been rather limited and practically none of them work for cassette. The majority of these autoboot file makers only allow one BASIC line to be executed, and often do not let the user see what that line is. The user is left totally in the dark while the computer executes the line. Limitations like these seriously restrict the usefulness of autoboot programs.

FAST FINGERS is different. FF works through the keyboard, fooling the computer into thinking that somebody is at the console typing each key. Let's say that you want to create an autoboot file which will automatically run a game program written in BASIC. If you have a disk drive, the way to do this manually would be to type the command RUN"D:GAME" when the READY prompt appears after turning on the computer. If you are using a cassette, you instead have to type the commands CLOAD and RUN. Either way, you must turn on the computer and wait for the READY prompt before you can start typing. As soon as the READY prompt appears, the computer is waiting for keyboard input. What FF does is generate keystrokes whenever the computer is waiting for keyboard input. If it generates the keystrokes to type commands like RUN and CLOAD, these commands will be executed. But FF is not limited to use in the immediate mode of BASIC. If the program contains an INPUT statement, perhaps to get a player's name, FF can type the name and press the

RETURN key in response. The BASIC program can't even tell whether the keystrokes are coming from a real live human or not. FF is not restricted to typing just one line; memory is the only limit to the amount of keystrokes FF can simulate. This means that it is possible to have FF type in and RUN a whole program! And FF is not limited to the BASIC cartridge, as it will also work with ASM/ED and PILOT. It can even call DUP or work in MEMO PAD mode! Yet the cost in memory is trivial. There is only a fifty six byte overhead, plus one byte for each keystroke.

The FAST FINGERS BOOT FILE MAKER program, used to create the autobooting file which contains the keystroke information, also has many fine features. This program lets you specify the memory location where the autoboot file first loads in, where the keystroke data is to be stored, and how quickly the keys are to be typed. The keystroke information can be entered immediately by the user, or can come from a disk or cassette file. What is so fascinating about FFBFMAKR is the fact that the keystroke information is taken directly from the user and stored in the autoboot file. For example, if you type in a line while using the boot file maker, make a mistake, and use the backspace key to correct it, all of these keystrokes will be generated at boot time, including the backspaces. The use of screen clear, tabbing, cursor controls, character and line insert and delete, lower case and even inverse is reproduced exactly.

HOW TO USE THE FAST FINGERS BOOT FILE MAKER

When you type in the FFBFMAKR program listed below, be very careful to type the all numbers correctly. You will also want to save a copy of the program before using. The explanation that follows shows how to use FFBFMAKR to automatically RUN a BASIC game program upon booting a cassette or disk.

Run the FFBFMAKR program. It will display a title and take a moment to initialize. Then the prompt "PLEASE ENTER INPUT DEVICE SPEC" will appear. Here you should type the device specification from which the text will come. Normally you would type either K:, C:, or D:filename. For purposes of explanation, let us assume you specify K: (or just K since there is no filename) for the keyboard.

Next the screen will be cleared and the prompt "READY" will be displayed, simulating the screen condition when the computer first enters BASIC. Don't be fooled - the program is still running. At this point, every keystroke which produces a legal ATASCII character is recorded. This means that bad key combinations, such as CTRL-4, and keys like SHIFT and CTRL which do not have ATASCII codes directly associated with them, are not valid. FFBFMAKR will, however, accept keystrokes for inverse and lower case.

If you are using cassette, type the following two lines. After typing CLOAD and pressing RETURN, press RETURN a second time. This corresponds to the pressing of the RETURN key which starts the program loading. You need type only one RETURN after the command RUN.

```
CLOAD (press RETURN)
(press RETURN again)
RUN (press RETURN)
```

If you are using a disk, you have just one line to type.

```
RUN"D:GAME (RETURN)
```

Remember that although the screen looks like BASIC, the FFBFMAKR program is still running. Immediate mode commands such as CLOAD will not be executed. If you make a typing mistake that you do not want to have seen at boot time, you will have to press BREAK and restart.

When you have entered all the keystrokes, terminate your input by pressing the keyboard's EOF (end of file) code, CTRL-3. A message will be printed, telling you how many

keystrokes were recorded. In the above examples, the keystroke totals would be twelve for both cassette and disk (remember that the RETURN and EOF keys are included).

BOOT FILE PARAMETERS

Now that the keystrokes have been recorded, a few questions must be answered before the binary file can be written. The first prompt asks whether you are using a cassette or disk. Type the correct answer (one letter, "C" or "D", is sufficient), and press RETURN.

If you are using a disk, before pressing RETURN, be sure you have put in the drive the disk to which the AUTORUN.SYS file will be written. When RETURN is pressed, the drive will be activated. Under certain circumstances, the prompt "APPEND?" may appear. For now, ignore it, and just press RETURN again.

When you power up the computer, the binary file is loaded into memory where code is executed to install FF. The next prompt asks for the loading address of the autoboot file, and displays the default value in parentheses. For cassette, the default value is 1792, and for disk it is 13312. A value other than the default can be entered by typing numbers before pressing RETURN. More information is given later about alternate values for the loading address. To select the default value, just press the RETURN key.

The "patch address" is the address at which the FF machine code routine and keystroke data is stored. The length of the "patch" is fifty six bytes plus one byte for each keystroke - the total was printed earlier. There must be at least this many contiguous unused bytes starting at the patch address. The default address for storing the patch is 1536, for page six. If FF is used to RUN a program which uses this area, it may be necessary to change the location where the patch is placed. Also, there is only enough room on page six to hold data for 200 keystrokes, and another section of memory will have to be used for larger amounts of data. As before, just press RETURN to choose the default value, which should be satisfactory in most cases.

The last prompt asks for the speed, with a scale from one to nine. This controls how quickly the keystrokes will be typed at boot time. The fastest speed, one, limits keystrokes to a speed of no more than sixty keys per second. At about one key every four seconds, speed nine is rather slow. The default is three.

After you press RETURN for the speed prompt, the program will write the binary boot file. If it is writing to the cassette, it will use the Short Inter Record Gap format, for the fastest possible loading. If the file is written to disk, the filename AUTORUN.SYS will be used. Note that if there was previously an AUTORUN.SYS file on the disk, it will be overwritten and replaced with the new file, unless it was locked, in which case an error 167 will be reported.

When FFBFMAKR is done, the READY prompt should appear. This time it is the real READY prompt. If you are using cassette, notice the position where the tape stopped. Now CSAVE the BASIC game program at this point. If you are using a disk, be sure that the disk contains the game program file and DOS.SYS.

BOOTING

To test the autoboot file, you will have to follow one of the two procedures described here, depending on whether you have cassette or disk. To start, turn the computer off, leaving the cartridge in place. If you have an 850 Interface, turn it off.

If the file was written to cassette, press and hold down the START key when you turn on the computer. You should hear the familiar single beep, indicating that the computer is ready for you to press PLAY on the program recorder (it is okay to release the START key once you hear the beep). The computer will load and execute the autoboot file at the location entered beforehand as the load address. The initialization part of file will move the patch to the location specified earlier as the patch address. After that, the computer will continue with the normal power up sequence. When BASIC prints READY and waits for input, FF will take over the keyboard. You should see the characters you previously typed, and hear the keyboard clicking as well. In short, the computer will be functioning just as if

you were rapidly pressing the same keys that you did when using FFBFMAKR, except that the keys on the keyboard do not move.

If you followed the examples and typed the commands CLOAD and RUN from FFBFMAKR, the following things will happen:

1. The command CLOAD will be typed on the screen, and the RETURN key will be pressed.
2. The computer will make the single beep to signal that the PLAY key should be pressed. Since this key should still be pressed from loading the autoboot file, you do not have to do anything.
3. The RETURN key will be typed, and the computer will start to load the BASIC program.
4. After the program is loaded, the command RUN will be typed, and the program will start.

The disk booting process is even easier. First the computer will load in DOS, then it will load and execute AUTORUN.SYS. After that, FF will take over the keyboard as soon as the READY prompt appears. The single command RUN"D:GAME will be enough to get the game going.

ANOTHER DEMONSTRATION

Here is another demonstration, to show you how FF can do more than just automatically RUN a program. When in the record mode of FFBFMAKR, type the following lines:

```
10 DIM NAME$(3)
20 ?"WHAT IS YOUR NAME";
```

Right after you press the RETURN key for line 20, use the cursor control keys to move the cursor up to line 10, tab over to the digit three in the dimension statement, do a control-insert, type the digits one and zero, and press the RETURN key. The DIM statement will be set to dimension a string ten characters long. Now move the cursor back to the line below line 20, and finish the typing.

```
30 I.NAME$
40 ? NAME$;" ";
50 G.40
L.
RUN
JEFF
```

Be sure to press RETURN after typing JEFF. Terminate input by pressing CTRL-3, specify cassette or disk, choose the defaults for load and patch addresses, use a speed of 4, and boot the cassette or disk. You will see the program be entered, the cursor will move up to correct line 10, the corrected program will be listed, it will begin executing, and the response of "JEFF" will be typed for the INPUT statement. You must admit, you've never seen your computer do anything like that before.

SPECIAL FEATURE FOR DISK USERS

If you are using a disk, a special feature in FFBFMAKR lets you combine AUTORUN.SYS files. After you press RETURN for the cassette/disk prompt, the program looks at the disk directory to see if an AUTORUN.SYS file already exists. If so, you will be asked whether you want to append the FF boot file to that AUTORUN.SYS. A one letter response of "Y" or "N" will do fine. Normally, you would respond "N" for "No", in which case the AUTORUN.SYS file already on the disk will be replaced with the FF boot file. But if you choose "Y", for "Yes", the AUTORUN.SYS will remain intact, and the FF boot file will be added at the end. When the computer boots, the original binary file will be loaded and executed, followed by the FF code. This file appending feature means that you can use FF

without having to give up the AUTORUN.SYS file. Of course, if no AUTORUN.SYS already exists on the disk, you would not be able to append the FF boot file to it, so the is no "APPEND?" prompt.

APPLICATIONS

You can always use FF like any other autobooting utility, to run one program. But even with a simple task like this, FF gives you some added flexibility. Users of DOS II are aware that an AUTORUN.SYS file is needed to load the RS232 handler for use with modem programs. With the APPEND option of FFBFMAKR, you can add the FF file to the end of the file which loads the RS232 handler from the 850 Interface. Now you can have your BASIC terminal program RUN automatically when you turn on the computer. You could even have FF enter some configuration commands, and if your modem has phone dialing capability, FF can type the phone number (just be sure to use a typing speed slower than the modem's baud rate). FF can be used in similar ways with other types of programs.

Let's say you want to demonstrate a program to somebody. The program might be menu oriented, requiring keyboard input - much like the DUP menu program. You can either tell the person to read the instructions, or manually show him how the program works. Or, if you use FF, you can have a whole sequence of keystrokes be typed when the program runs. The computer will demonstrate how the program works by choosing different menu items and answering the various questions, while the person sits back and watches. It would even be possible for FF to show how to use the FFBFMAKR program! And the nice thing is that the program to be demonstrated does not have to be modified in any way. Just provide the autobooting file with the program, and the program will demonstrate itself.

Here's one more excellent application for FF. This little item by itself makes FF worth the time it takes to type the FFBFMAKR program. Cassette librarians in user groups have a difficult time making many copies of cassettes. The copying process is very slow. To make cassette copying more convenient, use FF. First, take all the files which are to be copied to a cassette and save them on a disk. Then use FFBFMAKR to prepare a command sequence which consists of LOAD and CSAVE commands.

```
LOAD "D:GAME" (RETURN)
CSAVE (RETURN RETURN)
LOAD "D:PMDEMO" (RETURN)
CSAVE (RETURN RETURN)
LOAD "D:FFBFMAKR" (RETURN)
CSAVE (RETURN RETURN)
```

All you have to do is insert a cassette in the program recorder, press PLAY and RECORD, boot the disk, and watch the computer transfer the files for you. You can leave the computer completely unattended while it does the copying, although you may periodically want to glance at the screen to check for error messages. To get another copy, all you have to do is insert a different tape and reboot the computer. This is really a great time saver.

Many applications for FF have not even been thought of yet, because a utility like this has never been available before. Any time the computer needs input from the keyboard, that's when FF can be of use.

ALTERNATE INPUT DEVICE

When you run FFBFMAKR, the first prompt asks you which device to use for keystroke input. Usually, you choose K, for the keyboard, and start entering keystrokes from the keyboard. But editing can be tricky because every keystroke is recorded, even the editing keys. Because keyboard entry is not always convenient, you can specify that the input comes from a disk or cassette file. This means you can prepare a file using a text editor, save the file to cassette or disk, and use that file for keystroke information. As it gets

every character from the file, FFBFMAKR will display the character on the screen, and terminate input when the end of the file is reached. By keeping a copy of the original text file, small modifications can be made to it, and a new autoboot file created, without a whole lot of retyping required on your part.

If you LIST a BASIC program to cassette or disk and specify that file for keystroke input, the program will be entered line by line at boot time.

The ASM/ED cartridge can be used as a simple text editor. Enter the lines using the line numbers, but when you save the file, do not use the command LIST; use the PRINT command. PRINT does the same thing as LIST, except that it does not write out the line numbers.

## LIMITATIONS OF FAST FINGERS

FF will only work with ATASCII characters. Keys which are not associated with ATASCII codes are CTRL-1, CTRL-3 through CTRL-0, CTRL-?, CAPS/LOWR, the inverse key, all keys pressed with both SHIFT and CTRL, and the SHIFT and CTRL keys alone. If a character depends on the SHIFT or CTRL key being pressed simultaneously with another key, FF will handle that.

FF supports lower case by automatically generating a keycode for the CAPS/LOWR key the first time a lower case character is used. FF will remain in lower case mode until the end of the keycode data. This does not conflict with upper case characters because they are simulated as always having the SHIFT key depressed.

Inverse video is also supported, but there is a problem. Whenever the input data alternates between inverse and noninverse characters, FF will generate a keycode for the inverse key. For an undetermined reason, if the switch between normal and inverse video occurs a the beginning of a line in BASIC, the character after this keycode is sometimes not detected by the system at boot time when the higher speeds are used. The current solution is to either press the inverse key before pressing RETURN on the previous line, make the first inverse character be optional (such as a space), or use a slower speed.

FF can call the MEMO PAD, but there is no way it can return to BASIC once it is in that mode. FF can call DUP, but it will be de-activated when it returns from DUP to BASIC.

When entering text from K: in FFBFMAKR, only one press of the ESC key is needed to create the ESC character, and it will be translated into two ESC keystrokes. To have FFBFMAKR produce only one ESC keystroke, perhaps to have FF type "deferred" control codes at boot time, in line 310 of the program remove the two statements which read B$(L)=CHR$(28):L=L+1.

To force FF into generating a keycode for CTRL-3, which is the end of file key, change the Q=160 in line 350 of the program to Q=154 and press control comma whenever the CTRL-3 is desired.

Sorry, but FF cannot simulate a pressing of the BREAK key.

While FF has control of the keyboard, the keyboard interrupts are disabled, so keys like CTRL-1 and BREAK will not work. This is to prevent conflicts with the keycodes coming from FF. This can be changed using POKE to change POKMSK (location 16) and IRQEN (53774). Use POKE values of 64 to enable the keys, 128 to enable the BREAK key, 192 to enable both, and 0 to disable both. The keyboard is automatically re-enabled by DUP and the GRAPHICS command, and when FF runs out of keystroke data. You can, of course, manually stop FF by pressing System Reset.

## LOAD AND PATCH ADDRESSES

Three things are loaded at the load address; the FF initialization code which installs the patch, the patch routine, and the keystroke data. The initialization code is 57 bytes long, the patch is 56 bytes long, and the keystroke data takes one byte per keystroke. Altogether, the number of contiguous free bytes needed to load FF is 113 plus the keystroke count. An area of free memory this long must be found, and the starting address of this area is the load address specified in FFBFMAKR. However, once the code has been loaded and FF has been installed, this memory is no longer needed.

For cassette systems, free memory starts at page seven, which is decimal address 1792, the default value in FFBFMAKR. One occassion when page seven cannot be used is when the RS232 handler is loaded from the 850 Interface. In that situation, a higher memory address should be used, such as 4000.

On a disk system, the default vaue of 13312 was chosen because it is right after the point where DUP stops loading. A different value may have to be used if FF is appended to an AUTORUN.SYS file which needs the same area, or if a later version DOS needs that memory.

The memory reserved for the patch is needed until FF has generated all of its keystrokes. In most cases, the default value of 1536 (page six) will be okay. But page six can only contain 200 keystrokes, and if more are used, FFBFMAKR will print a warning after it prints the total keystroke count. Also, some programs use page six. Other suitable places for storing the patch would be the cassette buffer (there are 128 free bytes starting at 1024, which gives room for 128-56=72 keystrokes), in low memory (be sure to change MEMLO), or in the middle of free memory (such as just below the display list - but this is risky). To establish a safe area of memory, you may have to resort to the same tricks used to reserve memory for player/missile or character set buffers.

Cassette users should be aware that the RS232 handler uses page six on a cassette system, so another place will have to be used when you want an autobooting modem program. Also, on cassette only, the 850 Interface will not upload the RS232 handler to the computer at boot time if it is not turned off and back on before powering up the computer. You should turn the 850 off, back on again, and then turn on your computer.

## THE SECRET BEHIND FAST FINGERS

FF is one of the more unusual applications of the vertical blank interrupt. This is the interrupt that occurs every sixtieth of a second, when the television has finished drawing one frame. During this brief interval, the computer stops normal processing (like execution of a BASIC program) and bumps the real time clock, updates hardware and shadow registers, checks for attract mode, monitors the keyboard, and a does a few other "housekeeping" chores. Then it picks up the normal processing where it left off. Since this happens sixty times every second, the time spent processing the interrupt must be significantly less than a sixtieth of a second. Because the interrupt is so quick and happens so often, its effect on BASIC program execution is not noticeable. That is why vertical blank processing is said to be "transparent" to the system.

FF simply extends the keyboard monitoring of the vertical blank. The patch constantly checks location 764, also known as operating system variable CH. Whenever a key on the keyboard is pressed, the corresponding hardware keycode of that key is stored in CH. When the operating system is asked to input a character from the keyboard, it fetches the keycode from CH, converts it to an ATASCII character, and sets CH to 255, meaning that the key has been read. Thus CH is "empty" when it contains the value 255. If the operating system needs keyboard input and CH still contains a 255, it will wait until a key is pressed and the value in CH changes.

Location 764 is the key (ouch! a pun!) to FAST FINGERS. As soon as FF detects a 255 in CH, it retrieves the next keycode from the data buffer and stores it in CH, overwriting the 255. It will let that value stay there until the operating system reads it and sets CH back to a 255. Using this technique, FAST FINGERS is able to offer many advantages over other autoboot programs, while keeping the memory requirement down to fifty six bytes plus data.

Because FF does not generate the normal keyboard interrupt, keys pressed by FF will not cancel the attract mode. This does, however, provide the only way for a program to tell whether input is coming from FF or not. Have the program POKE location 77 (operating system variable ATRACT) with a value of 60 or so. Then have the program accept input from the keyboard. If the value in location 77 has not been reset to a very low number, FF must be generating the keystrokes.

## SUMMARY

Now that you have FAST FINGERS, you can do some amazing things with autobooting files. The largely unexplored area of autobooting cassettes is now open territory. Given the method by which FF is implemented, it is an easy matter to make your computer quickly and automatically execute any command sequence, run programs, or just have the cursor perform acrobatics. Turning on your computer was never so much fun!